CoAnim - Cryptologie : Partie 2

1- LE CHIFFREMENT PAR SUBSTITUTION :

Avec le chiffrement de César, les lettres sont toutes décalées de la même valeur. Il devient ainsi facile de mener une attaque en force brute. Une variante de cet algorithme consiste à substituer chaque lettre par une autre lettre, définie au hasard. Par exemple la lettre A est remplacée par la lettre G, la lettre B par la lettre D, la lettre Z par la lettre M, etc.

On appelle cette méthode le **chiffrement par substitution,** et dans ce cas la clé correspond au tableau de substitution de chaque lettre de l'alphabet.

Cette clé est une permutation de 26 éléments, il y a donc "26!" (factoriel de 26) clés possibles, soit 2 puissance 88, ou encore 10 puissance 27 (1 suivi de 27 zéros). Ce nombre gigantesque de clés possibles rend **l'attaque par force brute impossible à réaliser** dans la pratique sans moyens informatiques.

Exercice 1. : Soit le script de la fonction *genCle()* donné ci-dessous :

```
def genCle() :
    alphabet = "abcdefghijklmnopqrstuvwxyz"
    liste = []
    cle = ""
    for c in alphabet :
        liste.append(c)
    while liste != [] :
        i = randint(0,len(liste)-1)
        c = liste.pop(i)
        cle = cle + c
    return cle
```

Cette fonction n'a pas de paramètres, mais retourne un string de 26 caractères, constitué des 26 lettres de l'alphabet écrites dans un ordre aléatoire.

Par exemple, une première exécution de *genCle()* pourrait renvoyer :

```
>>> genCle()
'kiseworzfctmjydqlubnhgxvap'
```

une seconde exécution de *genCle()* pourrait renvoyer :

```
>>> genCle()
'ruatvzhjxindlsgwbmqcykfeop'
```

Pour comprendre ce qui se passe dans cette fonction, on rappelle ci-dessous l'existence de quelques fonctions natives de python, utiles ici :

Point Cours :

- La fonction *randint()* de la bibliothèque random , permet de générer un nombre aléatoire entier :

```
from random import randint
```

```
randint(0,26)
```

- La méthode *pop()* native de python, permet d'enlever un élément d'une liste et de le renvoyer en retour :

```
>>> liste = ['a','b','c']

>>> liste.pop(1)
'b'

>>> liste
['a', 'c']
```

⇨ Ecrire le code de la fonction *genCle()* dans un fichier nommé *chiffrementParSubstitution_VoreNom.py* .

Exercice 2. : Dans le même fichier *chiffrementParSubstitution_VoreNom.py*, compléter le script de la fonction *chiffrementSubstitution()* donnée ci-dessous :

```
def chiffrementSubstitution(texte,cle,code) :
    alphabet = "abcdefghijklmnopqrstuvwxyz"
    newTexte = ""
    ⇨ Script à compléter
    return newTexte

# Programme principal
cle = genCle()
print(f"clé utilisée : {cle}")
texteEnClair = "attaquez a l aube"

texteChiffre = chiffrementSubstitution(texteEnClair,cle,True)
print(texteChiffre)

texteDeChiffre = chiffrementSubstitution(texteChiffre,cle,False)
print(texteDeChiffre)
```

Cette fonction renvoie un texte crypté par substitution. La clé mise en argument aura été générée par la fonction *cleGen()* précédente.

L'exécution de ce code pourrait donner dans la console :

```
>>> (executing file "chiffrementParSubstitution.py")
clé utilisée : ruotvfiawsbpjxezlndchmqygk
rccrlhvk r p rhuv
attaquez a l aube
```

⇨ Uploader ce fichier *chiffrementParSubstitution_VoreNom.py* dans le répertoire Devoir sur U:/

CORRIGE

```
def chiffrementSubstitution(texte,cle,code) :
    alphabet = "abcdefghijklmnopqrstuvwxyz"
    newTexte = ""
    for c in texte :
        if c != " ":
            if code :
                i = alphabet.index(c)
                newC = cle[i]
            else :
                i = cle.index(c)
                newC = alphabet[i]
        else : newC = c
        newTexte = newTexte + newC
    return newTexte
```

Pour information : Ce chiffrement est moins sujet à une attaque par force brute que celui de César. Il reste par contre sujet à un autre type d'attaque : **l'analyse fréquentielle des lettres.** Dans la langue française, comme dans toutes les langues, certaines lettres sont plus fréquentes que d'autres. Le E par exemple est la lettre la plus courante, suivie par le S, alors que le W et le K sont les moins courantes. Avec le chiffrement par décalage, chaque lettre est toujours remplacée par la même lettre, en appliquant le décalage par une distance fixée. Ainsi, la fréquence d'apparition des lettres dans un message chiffré reste la même que pour le message clair.

En comptant le nombre d'occurrences de chaque lettre dans le message chiffré, vous voyez quelles lettres sont les plus fréquentes et lesquelles sont les moins fréquentes. Vous pouvez ainsi déduire que les lettres les plus fréquentes correspondent aux lettres les plus fréquentes de la langue française. Par exemple si le V est la lettre la plus fréquente dans le texte chiffré, vous pouvez déduire que la lettre E est permutée par la lettre V, etc.

Vous pouvez faire la même analyse fréquentielle sur les **digrammes,** c'est à dire les occurrences de 2 lettres de suite. En français, le digramme le plus fréquent est "ES", suivi de "LE". Avec un message chiffré suffisamment long, il y a forcément des motifs reconnaissables dans le texte chiffré qui permettent de le décrypter.

Ce type d'attaque s'appelle une **attaque par "texte chiffré seulement".** C'est la pire des attaques sur un système de chiffrement. Cela signifie qu'un attaquant peut retrouver le message en clair et la clé de déchiffrement simplement en observant les textes chiffrés, sans rien connaître de plus.